```sql
--DATA WRANGLING PROCESSES
--BELOW ARE THE DATA CLEANING STEPS USED IN THIS PROJECT
--1 STANDARDIZE DATE FORMAT(USING CONVERT)
--2 POPULATE THE PROPERTY ADDRESS FIELDS (USING SELF JOIN AND ISNULL)
--3 BREAKING ADDRESS INTO INDIVIDUAL COLUMN (ADDRESS, CITY, STATE) (USING PARSENAME)
--4 CHANGE Y AND N INTO YES AND NO IN THE 'SOLD AS VACANT' FIELD (USING UPDATE AND
CASE STATEMENTS)
--5 REMOVE ALL DUPLICATES (USING CTE, PARTITON BY AND ROW_NUMBER FUNCTIONS)
--6 DELETING UNUSED COLUMNS
---------------------------------------------------------------------------------------
-------------------------------------------
SELECT *
FROM HousingNashville;

---------------------------------------------------------------------------------------
---------------------------------------------
--1 STANDARDIZE DATE FORMAT(USING THE CONVERT FUNCTION)
SELECT CONVERT(DATE,SALEDATE)
FROM HousingNashville

--CREATE A NEW COLUMN ON THE TABLE USING THE ALTER COLUMN FUNCTION AND SAVE THE
CONVERTED DATE IN IT)
ALTER TABLE HousingNashville
ADD CONVERTEDSALEDATE DATE;

UPDATE HousingNashville
SET CONVERTEDSALEDATE = CONVERT(DATE,SALEDATE);

--REMOVE THE SALEDATE COLUMN FROM THE DATA
ALTER TABLE HousingNashville
DROP COLUMN SALEDATE;



---------------------------------------------------------------------------------------
------------------------------------------------------
--2 POPULATE THE PROPERTY ADDRESS FIELDS
-- FIRST WE DO A SELF JOIN OF THE TABLE TO ITSELF BUT WHERE PROPERTYADDRESS COLUMN IS
NOTNULL, THEN USE ISNULL FUNCTION THE POPULATE THE NULL FIELD
SELECT A.PARCELID, A.PROPERTYADDRESS, B.PARCELID, B.PROPERTYADDRESS,
ISNULL(A.PROPERTYADDRESS,B.PROPERTYADDRESS)
FROM HousingNashville AS A JOIN
HousingNashville AS B
ON A.PARCELID = B.PARCELID AND
A.UNIQUEID != B.UNIQUEID
WHERE A.PROPERTYADDRESS IS NULL

--USING THE UPDATE FUNCTION TO UPDATE THE TABLE AND REPLACE ALL NULL VALUES
UPDATE A
SET PROPERTYADDRESS = ISNULL(A.PROPERTYADDRESS,B.PROPERTYADDRESS)
FROM HousingNashville AS A JOIN
HousingNashville AS B
ON A.PARCELID = B.PARCELID AND
A.UNIQUEID != B.UNIQUEID
WHERE A.PROPERTYADDRESS IS NULL

---------------------------------------------------------------------------------------
-------------------------------------------------
--3 BREAKING ADDRESS INTO INDIVIDUAL COLUMN (ADDRESS, CITY, STATE)
-- THIS CAN BE DONE USING SUBSTRING OR PARSENAME FUNCTION, I CHOOSE TO USE THE
PARSENAME FUNCTION,
--BUT FIRST I'LL HAVE TO CHANGE THE COMMA DELIMETER TO FULLSTOP BEFORE USING THE
PARSENAME FUNCTION
SELECT PROPERTYADDRESS
FROM HOUSINGNASHVILLE

SELECT PARSENAME(REPLACE(PROPERTYADDRESS, ',' , '.'), 2),
PARSENAME(REPLACE(PROPERTYADDRESS, ',' , '.'), 1)
FROM HOUSINGNASHVILLE

ALTER TABLE HOUSINGNASHVILLE
ADD PROPERTYADD NVARCHAR(255)

UPDATE HOUSINGNASHVILLE
SET PROPERTYADD = PARSENAME(REPLACE(PROPERTYADDRESS, ',' , '.'), 2)

ALTER TABLE HOUSINGNASHVILLE
ADD PROPERTYCITY NVARCHAR(255)

UPDATE HOUSINGNASHVILLE
SET PROPERTYCITY = PARSENAME(REPLACE(PROPERTYADDRESS, ',' , '.'), 1)

SELECT *
FROM HousingNashville

--FOR OWNER ADDRESS FIELD
SELECT OWNERADDRESS
FROM HousingNashville

SELECT PARSENAME(REPLACE(OWNERADDRESS, ',', '.'),3),
PARSENAME(REPLACE(OWNERADDRESS, ',', '.'),2),
PARSENAME(REPLACE(OWNERADDRESS, ',', '.'),1)
FROM HousingNashville

ALTER TABLE HousingNashville
ADD OWNERADD NVARCHAR(255)

UPDATE HousingNashville
SET OWNERADD = PARSENAME(REPLACE(OWNERADDRESS, ',', '.'),3)

ALTER TABLE HousingNashville
ADD OWNERCITY NVARCHAR(255)

UPDATE HousingNashville
SET OWNERCITY = PARSENAME(REPLACE(OWNERADDRESS, ',', '.'),2)

ALTER TABLE HousingNashville
ADD OWNERSTATE NVARCHAR(255)

UPDATE HousingNashville
SET OWNERSTATE = PARSENAME(REPLACE(OWNERADDRESS, ',', '.'),1)

--DROPPING THE PREVIOUS ADDRESS TABLES
ALTER TABLE HousingNashville
DROP COLUMN PROPERTYADDRESS

ALTER TABLE HousingNashville
DROP COLUMN OWNERADDRESS

SELECT *
FROM HousingNashville
```

```sql
--------------------------------------------------------------------------------------
----------------------------------------------------------
--4 CHANGE Y AND N INTO YES AND NO IN THE 'SOLD AS VACANT' FIELD
-- USING A CASE WHEN FUNCTION TO CHANGE Y AND N AD THEN UPDATE THE TABLE AND THE
COLUMN WITH A SET FUNCTION.
SELECT SOLDASVACANT
, CASE WHEN SOLDASVACANT = 'Y' THEN 'YES'
          WHEN SOLDASVACANT = 'N' THEN 'NO'
          ELSE SOLDASVACANT
          END
FROM HousingNashville

UPDATE HousingNashville
SET SOLDASVACANT = CASE WHEN SOLDASVACANT = 'Y' THEN 'YES'
          WHEN SOLDASVACANT = 'N' THEN 'NO'
          ELSE SOLDASVACANT
          END

SELECT *
FROM HousingNashville

--------------------------------------------------------------------------------------
----------------------------------------------------
--5 REMOVE ALL DUPLICATES
--Writing a CTE with a windows function PARTITION BY using ROW_NUMBER function over
columns with a likely duplicate and
--then filtering where (dup_rows) is gretaer than 1 and then delete


WITH DUPROWSCTE AS(
SELECT *,
       ROW_NUMBER() OVER(
       PARTITION BY PARCELID,
                             PROPERTYADD,
                             PROPERTYCITY,
                             SALEPRICE,
                             CONVERTEDSALEDATE,
                             LEGALREFERENCE
                             ORDER BY
                                   UNIQUEID
                                   ) AS DUP_ROWS

FROM HousingNashville
)
SELECT *
FROM DUPROWSCTE
WHERE DUP_ROWS > 1
ORDER BY PROPERTYADD


--TO DELETE THE DUPLICATES

WITH DUPROWSCTE AS(
SELECT *,
       ROW_NUMBER() OVER(
       PARTITION BY PARCELID,
                             PROPERTYADD,
                             PROPERTYCITY,
                             SALEPRICE,
                             CONVERTEDSALEDATE,
                             LEGALREFERENCE
                             ORDER BY
```

```
                                                   UNIQUEID
                                                   ) AS DUP_ROWS

FROM HousingNashville
)
DELETE
FROM DUPROWSCTE
WHERE DUP_ROWS > 1


select *
from HousingNashville

-----------------------------------------------------------------------------------------
------------------------------------------------------------------
--6 DELETING UNUSED COLUMNS
--This is only advised to be done in views and not actual tables in the database.
-- The unused columns were deleted in the 3rd analysis.
```